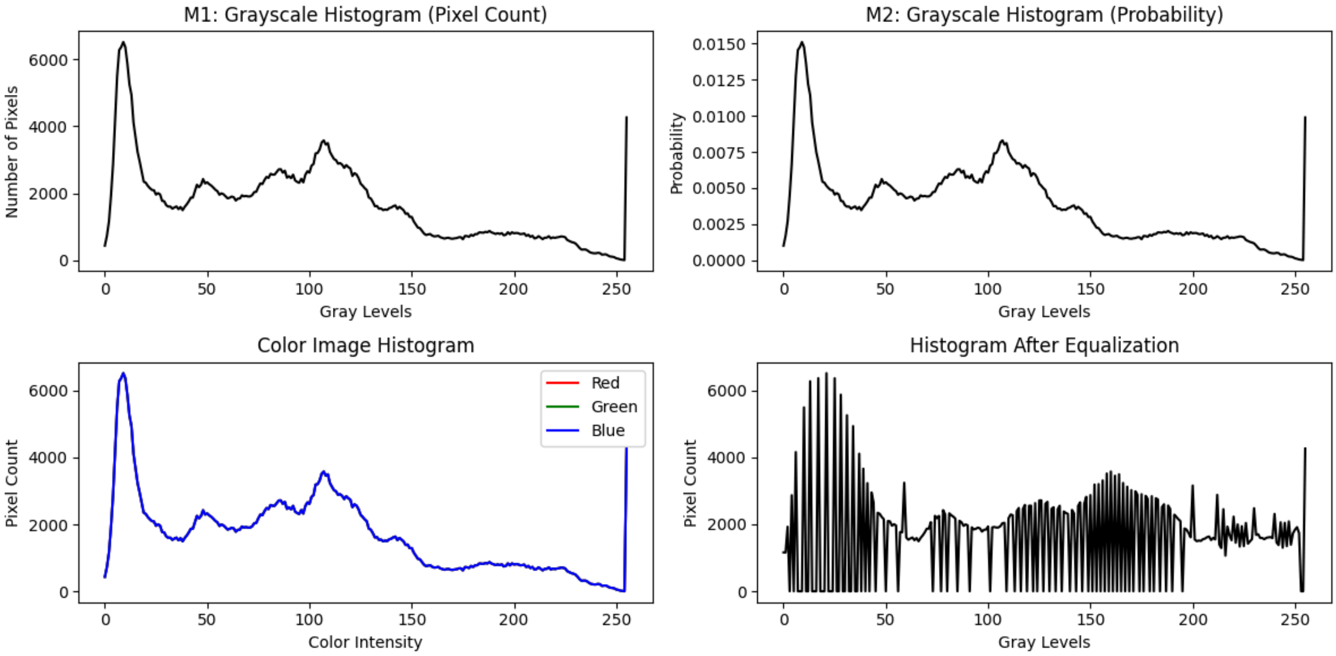
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
from google.colab import files

# Upload an image
uploaded = files.upload()
image_name = list(uploaded.keys())[0]

# Read the color image
color_image = cv2.imread(image_name)

# Convert from BGR to RGB (Matplotlib expects RGB format)
color_image_rgb = cv2.cvtColor(color_image, cv2.COLOR_BGR2RGB)

# Convert color image to grayscale
gray_image = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)

# **Compute Histogram Values**
# Color Image Histogram (for R, G, B channels)
color_hist_r = cv2.calcHist([color_image], [2], None, [256], [0, 256])
color_hist_g = cv2.calcHist([color_image], [1], None, [256], [0, 256])
color_hist_b = cv2.calcHist([color_image], [0], None, [256], [0, 256])

# Grayscale Histogram
gray_hist = cv2.calcHist([gray_image], [0], None, [256], [0, 256])

# Normalize histogram for M2 (probability distribution)
gray_hist_prob = gray_hist / gray_hist.sum()

# **Histogram Visualization**
plt.figure(figsize=(12, 6))

# **M1: Intensity vs Pixel Count (Grayscale)**
plt.subplot(2, 2, 1)
plt.plot(gray_hist, color="black")
plt.title("M1: Grayscale Histogram (Pixel Count)")
plt.xlabel("Gray Levels")
plt.ylabel("Number of Pixels")

# **M2: Intensity vs Probability (Grayscale)**
plt.subplot(2, 2, 2)
plt.plot(gray_hist_prob, color="black")
plt.title("M2: Grayscale Histogram (Probability)")
plt.xlabel("Gray Levels")
plt.ylabel("Probability")

# **Color Histogram (RGB)**
plt.subplot(2, 2, 3)
plt.plot(color_hist_r, color="red", label="Red")
plt.plot(color_hist_g, color="green", label="Green")
plt.plot(color_hist_b, color="blue", label="Blue")
plt.title("Color Image Histogram")
plt.xlabel("Color Intensity")
plt.ylabel("Pixel Count")
plt.legend()

# **Histogram Equalization**
equalized_gray = cv2.equalizeHist(gray_image)
equalized_hist = cv2.calcHist([equalized_gray], [0], None, [256], [0, 256])

plt.subplot(2, 2, 4)
plt.plot(equalized_hist, color="black")
plt.title("Histogram After Equalization")
plt.xlabel("Gray Levels")
plt.ylabel("Pixel Count")

plt.tight_layout()
plt.show()

# **Display Original, Grayscale & Enhanced Image**
cv2_imshow(color_image)
cv2_imshow(gray_image)
cv2_imshow(equalized_gray)

# Save Equalized Image
equalized_image_path = "contrast_enhanced.png"
cv2.imwrite(equalized_image_path, equalized_gray)
files.download(equalized_image_path)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Screenshot 2025-01-30 114305.png to Screenshot 2025-01-30 114305.png
```



```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
from google.colab import files

# Upload an image
uploaded = files.upload()
image_name = list(uploaded.keys())[0]

# Read the image in grayscale
image = cv2.imread(image_name, cv2.IMREAD_GRAYSCALE)

# **Step 1: Compute the FFT**
dft = np.fft.fft2(image)
dft_shift = np.fft.fftshift(dft)  # Shift zero frequency to the center
magnitude_spectrum = 20 * np.log(np.abs(dft_shift))  # Compute magnitude

# **Step 2: Compute the Inverse FFT**
idft_shift = np.fft.ifftshift(dft_shift)  # Inverse shift
reconstructed_image = np.fft.ifft2(idft_shift)  # Inverse FFT
reconstructed_image = np.abs(reconstructed_image)  # Get real values

# **Step 3: Rotate the Image**
angle = 45  # Rotate by 45 degrees
(h, w) = image.shape[:2]
center = (w // 2, h // 2)
rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
```

```
rotated_image = cv2.warpAffine(image, rotation_matrix, (w, h))

# Compute FFT of Rotated Image
dft_rotated = np.fft.fft2(rotated_image)
dft_shift_rotated = np.fft.fftshift(dft_rotated)
magnitude_spectrum_rotated = 20 * np.log(np.abs(dft_shift_rotated))

# **Step 4: Display Results**
plt.figure(figsize=(12, 8))

# Original Image
plt.subplot(2, 3, 1)
plt.imshow(image, cmap="gray")
plt.title("Original Image")
plt.axis("off")

# Magnitude Spectrum of Original Image
plt.subplot(2, 3, 2)
plt.imshow(magnitude_spectrum, cmap="gray")
plt.title("Magnitude Spectrum (FFT)")
plt.axis("off")

# Reconstructed Image (IFFT)
plt.subplot(2, 3, 3)
plt.imshow(reconstructed_image, cmap="gray")
plt.title("Reconstructed Image (IFFT)")
plt.axis("off")

# Rotated Image
plt.subplot(2, 3, 4)
plt.imshow(rotated_image, cmap="gray")
plt.title(f"Rotated Image ({angle}°)")
plt.axis("off")

# Magnitude Spectrum of Rotated Image
plt.subplot(2, 3, 5)
plt.imshow(magnitude_spectrum_rotated, cmap="gray")
plt.title("FFT of Rotated Image")
plt.axis("off")

# Compare Spectra
plt.subplot(2, 3, 6)
plt.imshow(np.abs(magnitude_spectrum - magnitude_spectrum_rotated), cmap="gray")
plt.title("Difference in Spectra")
plt.axis("off")

plt.tight_layout()
plt.show()

# **Step 5: Save and Download Result Images**
cv2.imwrite("fft_magnitude.png", magnitude_spectrum)
cv2.imwrite("ifft_reconstructed.png", reconstructed_image)
cv2.imwrite("rotated_image.png", rotated_image)
cv2.imwrite("fft_rotated_magnitude.png", magnitude_spectrum_rotated)

# Download results
files.download("fft_magnitude.png")
files.download("ifft_reconstructed.png")
files.download("rotated_image.png")
files.download("fft_rotated_magnitude.png")
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Screenshot 2025-01-30 114305.png to Screenshot 2025-01-30 114305 (1).png
```